**客户关系管理系统**

[一、CRM系统简介 2](#_Toc36201228)

[1. CRM 2](#_Toc36201229)

[2. 系统功能图 2](#_Toc36201230)

[二、项目搭建 5](#_Toc36201231)

[1. 新建项目\*\*\*\_crm 5](#_Toc36201232)

[2. 配置数据源 7](#_Toc36201233)

[3. 使用Mybatis-plus代码生成器（生成实体类、Mapper接口、service接口及其实现类） 7](#_Toc36201234)

[三、模块开发 10](#_Toc36201235)

[1. 员工信息模块开发 10](#_Toc36201236)

[2. 客户管理模块开发 10](#_Toc36201237)

# 一、CRM系统简介

## 1. CRM

（1）CRM（Customer Relationship Management），客户关系管理系统

## 2. 系统功能图

**总体功能：**每个员工都可以登录这个系统，上传自己和客户的信息，将自己与客户的业务信息记录在该系统上面。（CRM系统一般是销售部在使用， 如果你想做一个所有部门都用的系统，那你应该做OA系统[办公自动化（Office Automation，简称OA）]）

【**方法：**要做一个系统，你可以在开源代码社区搜索类似项目，然后研究别人如何设计】

**角色权限：**

**总经办、主管：**员工的信息管理，客户信息管理、业务信息管理，数据统计**普通员工：**客户信息管理、业务信息管理，数据统计（自己的数据）

员工管理模块：

1. 员工(employee)信息的增删查改，员工属于某个角色

2. 角色（role）信息增删查改，角色属于某个部门

3. 部门（department）的增删改查

4. 权限（permission）的增删改查，每个角色拥有不同的权限

【employee、role、department、permission，emp\_role、dept \_role，role\_perm】

【员工表，角色表，部门表，权限表，员工角色关系表，角色部门关系表表，角色权限关系表】

【**最难是：将系统的功能分析划分出来，然后建立完整关系的数据表**，**一开始就要将数据表搞定，不要中途就修改数据表**】

客户管理模块：

1. 客户(customer)，（客户是属于某个销售员 ，多对一，员工登录后只能看到自己客户）

2. 跟踪（contact），销售员跟踪联系客户

3. 正式客户（is\_orders **0** 未曾下单，**1** 下过单），下单之后成为正式客户（即会员）

4. 客户池，查看所有的客户（只有主管或者超级管理员才有权限查看到所有客户）

customer customer\_id, customer\_name,sex, telphone, company ,address, emp\_id,is\_orders

客户id,姓名,性别,手机号码,公司,公司地址,接待的员工id，是否下过订单

contact contact\_id,way, content,customer\_id,emp\_id

联系id，联系途径（电话、qq、微信），聊天内容，客户id，员工id

业务管理模块：

1. 业务(business)，公司的业务（批发衣服、批发大米）

2. 订单（orders），客户下订单（这里可以关联订单系统，获取数据，或者从别的系统导出Excel数据，导入到这个功能）

business business\_id,busubess\_name, head, telphone,description

业务id，业务名称，负责人，电话号码，描述

orders orders\_id,business\_id, total\_price,customer\_id

订单id，业务id，共价，客户id

数据统计模块：

1. 统计数据 （查询员工人数、订单等等）

2. 生成统计表

【各种查询，数量、今天的数量，季度的数量】

-- 员工表

CREATE TABLE employee(

emp\_id INT PRIMARY KEY AUTO\_INCREMENT,

emp\_name VARCHAR(50) NOT NULL,

pwd VARCHAR(255) NOT NULL,

salt VARCHAR(255) NOT NULL,

age INT NOT NULL,

sex VARCHAR(3) NOT NULL,

phone VARCHAR(20) NOT NULL,

address VARCHAR(255) NOT NULL,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL

)ENGINE INNODB CHARSET= UTF8;

-- 角色表

CREATE TABLE role(

role\_id INT PRIMARY KEY AUTO\_INCREMENT,

role\_name VARCHAR(50) NOT NULL,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL

)ENGINE INNODB CHARSET= UTF8;

-- 用户角色关系表

CREATE TABLE emp\_role(

emp\_id INT,

role\_id INT,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL,

PRIMARY KEY (emp\_id,role\_id)

)ENGINE INNODB CHARSET= UTF8;

-- 部门表

CREATE TABLE department(

dept\_id INT PRIMARY KEY AUTO\_INCREMENT,

dept\_name VARCHAR(50) NOT NULL,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL

)ENGINE INNODB CHARSET= UTF8;

-- 部门角色关系表

CREATE TABLE dept\_role(

dept\_id INT,

role\_id INT,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL,

PRIMARY KEY (dept\_id,role\_id)

)ENGINE INNODB CHARSET= UTF8;

-- 权限表

CREATE TABLE permission(

perm\_id INT PRIMARY KEY AUTO\_INCREMENT,

perm\_name VARCHAR(50) NOT NULL,

url VARCHAR(255) NOT NULL,

permission VARCHAR(255) NOT NULL,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL

)ENGINE INNODB CHARSET= UTF8;

-- 角色权限关系表

CREATE TABLE role\_permission(

role\_id INT,

perm\_id INT,

create\_time DATETIME NOT NULL,

update\_time DATETIME NOT NULL,

is\_del INT NOT NULL,

PRIMARY KEY (role\_id, perm\_id)

)ENGINE INNODB CHARSET= UTF8;

# 二、项目搭建

## 1. 新建项目\*\*\*\_crm

*<!-- thymeleaf模板引擎 -->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
 </**dependency**>  
 *<!-- shiro -->* <**dependency**>  
 <**groupId**>org.apache.shiro</**groupId**>  
 <**artifactId**>shiro-spring</**artifactId**>  
 <**version**>1.4.0</**version**>  
 </**dependency**>  
 *<!-- thymeleaf模板引擎和shiro框架的整合 -->* <**dependency**>  
 <**groupId**>com.github.theborakompanioni</**groupId**>  
 <**artifactId**>thymeleaf-extras-shiro</**artifactId**>  
 <**version**>2.0.0</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
  
 *<!-- 热部署-->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-devtools</**artifactId**>  
 <**scope**>runtime</**scope**>  
 <**optional**>true</**optional**>  
 </**dependency**>  
  
 *<!-- mybatisPlus 核心库 -->* <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>3.1.0</**version**>  
 </**dependency**>  
 *<!-- 引入阿里数据库连接池 -->* <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>druid</**artifactId**>  
 <**version**>1.1.6</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**scope**>runtime</**scope**>  
 </**dependency**>  
 *<!-- 小辣椒-->* <**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**optional**>true</**optional**>  
 </**dependency**>  
  
 *<!-- Swagger 2-->* <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 <**version**>2.9.2</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 <**version**>2.9.2</**version**>  
 </**dependency**>  
  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 <**exclusions**>  
 <**exclusion**>  
 <**groupId**>org.junit.vintage</**groupId**>  
 <**artifactId**>junit-vintage-engine</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
 </**dependency**>  
</**dependencies**>  
  
<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

## 2. 配置数据源

（1）在application.yml中配置Tomcat端口号，以及数据源

（2）在application.yml配置mybatis-plus

（3）创建与配置logback-spring.xml

**搞定之后可以写个查询，测试效果**

## 3. 使用Mybatis-plus代码生成器（生成实体类、Mapper接口、service接口及其实现类）

（1）生成器代码

**import** com.baomidou.mybatisplus.core.exceptions.MybatisPlusException;  
**import** com.baomidou.mybatisplus.core.toolkit.StringPool;  
**import** com.baomidou.mybatisplus.core.toolkit.StringUtils;  
**import** com.baomidou.mybatisplus.generator.AutoGenerator;  
**import** com.baomidou.mybatisplus.generator.InjectionConfig;  
**import** com.baomidou.mybatisplus.generator.config.\*;  
**import** com.baomidou.mybatisplus.generator.config.po.TableInfo;  
**import** com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;  
**import** com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.Scanner;  
  
**public class** CodeGenerator {  
  
 */\*\*  
 \* <p>  
 \* 读取控制台内容  
 \* </p>  
 \*/* **public static** String scanner(String tip) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 StringBuilder help = **new** StringBuilder();  
 help.append(**"请输入"** + tip + **"："**);  
 System.***out***.println(help.toString());  
 **if** (scanner.hasNext()) {  
 String ipt = scanner.next();  
 **if** (StringUtils.*isNotEmpty*(ipt)) {  
 **return** ipt;  
 }  
 }  
 **throw new** MybatisPlusException(**"请输入正确的"** + tip + **"！"**);  
 }  
  
 **public static void** main(String[] args) {  
 *// 代码生成器* AutoGenerator mpg = **new** AutoGenerator();  
  
 *// 全局配置* GlobalConfig gc = **new** GlobalConfig();  
 **final** String projectPath = System.*getProperty*(**"user.dir"**);  
 gc.setOutputDir(projectPath + **"/src/main/java"**);  
 gc.setAuthor(**"作者名称"**);  
 gc.setBaseResultMap(**true**);  
 gc.setBaseColumnList(**true**);  
 *// 是否打开输出目录 默认为true* gc.setOpen(**false**);  
 mpg.setGlobalConfig(gc);  
  
 *// 数据源配置* DataSourceConfig dsc = **new** DataSourceConfig();  
 dsc.setUrl(**"jdbc:mysql://localhost:3306/czx\_crm?useUnicode=true&useSSL=false&characterEncoding=utf8"**);  
 dsc.setDriverName(**"com.mysql.jdbc.Driver"**);  
 dsc.setUsername(**"root"**);  
 dsc.setPassword(**"root"**);  
 mpg.setDataSource(dsc);  
  
 *// 包配置* **final** PackageConfig pc = **new** PackageConfig();  
 *// pc.setModuleName(scanner("模块名"));* pc.setParent(**"com.hopu.crm"**);  
 mpg.setPackageInfo(pc);  
  
 *// 自定义配置* InjectionConfig cfg = **new** InjectionConfig() {  
 @Override  
 **public void** initMap() {  
 *// to do nothing* }  
 };  
  
 *// 如果模板引擎是 freemarker* String templatePath = **"/templates/mapper.xml.ftl"**;  
 *// 如果模板引擎是 velocity  
 // String templatePath = "/templates/mapper.xml.vm";  
  
 // 自定义输出配置* List<FileOutConfig> focList = **new** ArrayList<FileOutConfig>();  
 *// 自定义配置会被优先输出* focList.add(**new** FileOutConfig(templatePath) {  
 @Override  
 **public** String outputFile(TableInfo tableInfo) {  
 *// 自定义输出文件名 ， 如果你 Entity 设置了前后缀、此处注意 xml 的名称会跟着发生变化！！* **return** projectPath + **"/src/main/resources/mapper/"** + tableInfo.getEntityName() + **"Mapper"** + StringPool.***DOT\_XML***;  
 }  
 });  
 cfg.setFileOutConfigList(focList);  
 mpg.setCfg(cfg);  
  
 *// 配置模板* TemplateConfig templateConfig = **new** TemplateConfig();  
  
 *// 配置自定义输出模板  
 // 指定自定义模板路径，注意不要带上.ftl/.vm, 会根据使用的模板引擎自动识别  
 // templateConfig.setEntity("templates/entity2.java");  
 // templateConfig.setService();  
 // templateConfig.setController();* templateConfig.setXml(**null**);  
 mpg.setTemplate(templateConfig);  
  
 *// 策略配置* StrategyConfig strategy = **new** StrategyConfig();  
 strategy.setNaming(NamingStrategy.***underline\_to\_camel***);  
 strategy.setColumnNaming(NamingStrategy.***underline\_to\_camel***);  
 *// strategy.setSuperEntityClass("com.fame.common.BaseEntity");* strategy.setEntityLombokModel(**true**);  
 strategy.setRestControllerStyle(**true**);  
 *// strategy.setSuperControllerClass("com.fame.common.BaseController");* strategy.setInclude(*scanner*(**"表名，多个英文逗号分割"**).split(**","**));  
 strategy.setSuperEntityColumns(**"id"**);  
 *// strategy.setControllerMappingHyphenStyle(true);* strategy.setTablePrefix(**"sys"**);  
 mpg.setStrategy(strategy);  
 mpg.setTemplateEngine(**new** FreemarkerTemplateEngine());  
 mpg.execute();  
 }  
}

# 三、模块开发

## 1. 员工信息模块开发

（1）查询员工（员工id、员工姓名，年龄、性别、住址、电话、角色、部门、创建时间，修改时间）

编辑EmployeeMapper.xml

查询员工信息（分页）

1. 分页配置类

2. 设置Mapper需要分页的方法

IPage<Employee> selectList();

## 2. 客户管理模块开发

（1）查询客户 根据shiro中当前用户emp\_id查询客户表

（2）跟踪（contact）根据shiro中当前用户emp\_id查询联系表

（3）正式客户（is\_orders 0 未曾下单，1 下过单），根据shiro中当前用户emp\_id查询客户表，并且is\_orders=1

（3）客户池（只有超级管理员有权限）查询所有客户

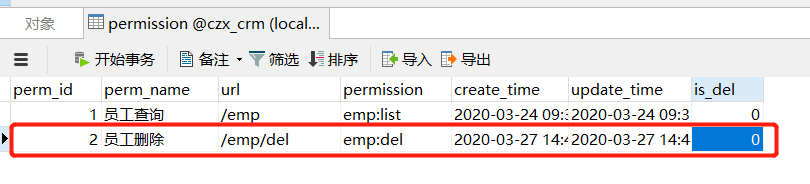
# 四、shiro的权限配置

## 1. 在shiro配置中，配置过滤的url及允许的权限表达式

*//待会需要给url设置权限 （写完realm的授权之后）*filterMap.put(**"/emp"**, **"perms[emp:list]"**);  
filterMap.put(**"/emp/del"**, **"perms[emp:del]"**);  
  
filterMap.put(**"/customer/pool"**, **"perms[customer:pool]"**);

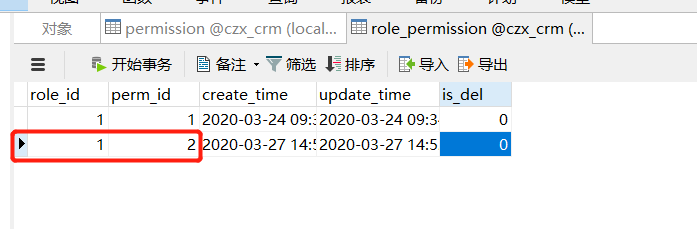
## 2. 在权限功能添加权限

【因为还没那个权限的增删查改功能，在permission表增加】



3. 给角色赋予权限

【因为还没那个角色的增删查改功能，在role\_permission表增加，给角色添加一个权限】



4. 在前端页面中配置角色判断



有权限的情况：



没有权限的情况：不显示那个li标签代码块



# 五、问题

## 1.数据库问题

报错:ERROR 1055 (42000): Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'work\_ad.api\_community\_pic.id' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql\_mode=only\_full\_group\_by

解决办法：

sql\_mode='STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

select distinct e.emp\_id,emp\_name,pwd, salt,age,sex,phone,address,e.create\_time,e.update\_time,e.is\_del,

r.role\_id,r.role\_name,p.perm\_id,p.perm\_name,p.permission,p.url,d.dept\_id,d.dept\_name

from employee e

LEFT JOIN emp\_role er on e.emp\_id = er.emp\_id -- 员工和角色的关系

LEFT JOIN role r on r.role\_id = er.role\_id

LEFT JOIN role\_permission rp ON r.role\_id= rp.role\_id

LEFT JOIN permission p ON p.perm\_id= rp.perm\_id

LEFT JOIN dept\_role dr ON r.role\_id = dr.role\_id

LEFT JOIN department d ON d.dept\_id = dr.dept\_id

WHERE e.is\_del = 0 GROUP BY e.emp\_id